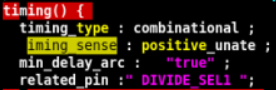
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Nguyen Le Thao Vy | No. | 3341 | Div/Dept | DSD/ACD/ACT3 | Job  Date：Jan 09th 2024  Title | Engineer |
| Please tick  the period | First Month | □W1 □W2 □W3 □W4 | | | | | |
| Second Month | □W1 □W2 🗹W3 □W4 | | | | | |
| Third Month | □W1 □W2 □W3 □W4 | | | | | |

1. The weekly report aims to help accelerate member’s workspace integration and should be reviewed by mentor on the last working day of the week.
2. The new weekly report should be reviewed and signed by mentor and direct supervisor.

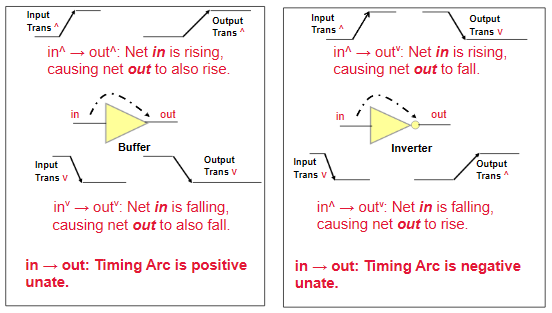
|  |
| --- |
| Work Experience Record |
| 1. Please describe the tasks and achievements you learned/executed:   All of the tasks I learned during this first week are based on the training plan of 2023, below are a brief description of topics and tasks I was working on:   * Topics of interest:   + STA;   + MBIST simulation;   + Testchip workflow. * Tasks that I was working on:   + Read materials – self-learn;   + Work on testchip FSFGFS016A. * Outcomes:   + Learnt basic concepts that aid in working process;   + Got familiar with running design kit and working in project. |
| 1. What are the problems encountered this week? Any actions taken? Any help needed?  * Problem encountered:   + Being unfamiliar with the testchip workflow. * Actions taken:   + Went through the flow in [FTV\_ACD\_TestChip\_Flow\_202303](file:///T:\FTV\DSD\DSD_ACD\ACT4\02_Internal_Tech_Shared\03_Testchip\FTV_ACD_TestChip_Flow_202303) under mentor’s guidance. |
| 1. What are the tasks for next week? Any preparation needed in advance?  * Work on assigned project; * Make time for completing items listed in training plan. |
|  |

|  |  |  |
| --- | --- | --- |
| Name  (Date) | Mentor | Direct Supervisor |
| Nguyen Le Thao Vy  (2024/01/21) | (Signature/Date) | (Signature/Date) |

***Timing sense / Unateness***

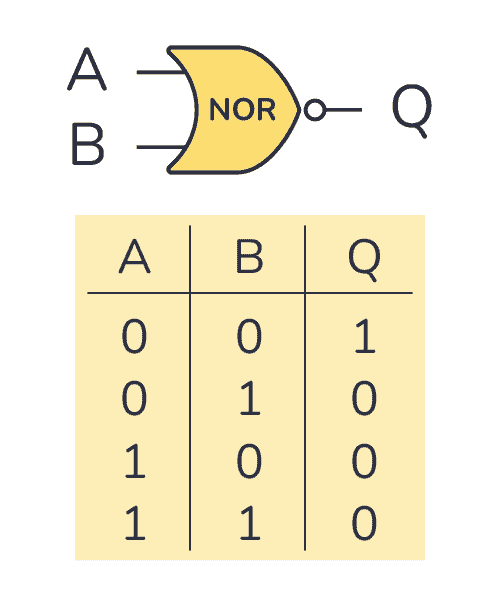
Timing sense / unateness is an attribute showing how a timing arc’s output will change with respect to different types of transitions on its input.

There are 3 types of unateness:

* Positive unate: A timing arc is positive unate if generally its output signal transition (rise or fall) is *the same* as input, i.e rising input 🡪 rising / unchanged output, falling input 🡪 falling / unchanged output.
* Negative unate: A timing arc is negative unate if generally its output signal transition (rise or fall) is *opposite* of input, i.e rising input 🡪 falling / unchanged output, falling input 🡪 rising / unchanged output.
* Non unate: For a non unate timing arc, changes in its output signal can not be determined based on the direction of an input value, meaning that the output value is not dependent on a single input value.

Example of determining unateness:

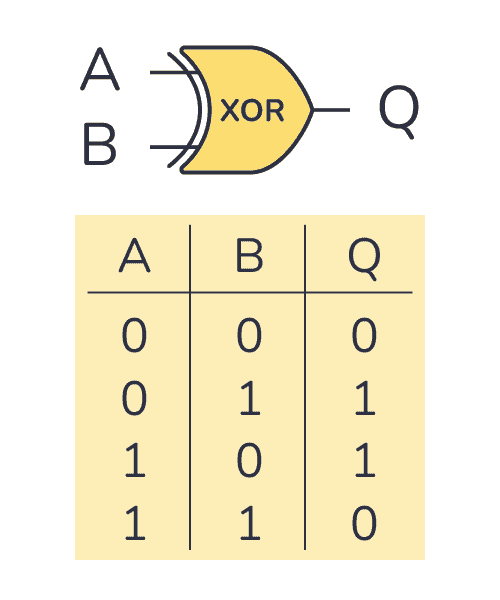
* + NOR gate 🡪 arcs are negative unate

Consider the rising input signals:

* + - A is rising (0 🡪 1):
      * if B = 0, Q is falling (1 🡪 0);
      * if B = 1, Q remains unchanged (0).
    - B is rising (0 🡪 1):
      * if A = 0, Q is falling (1 🡪 0);
      * if A = 1, Q remains unchanged (0).

Consider falling input signals:

* + - A is falling (1 🡪 0):
      * if B = 0, Q is rising (0 🡪 1);
      * if B = 1, Q remains unchanged (0).
    - B is falling (1 🡪 0):
      * if A = 0, Q is rising (0 🡪 1);
      * if A = 1, Q remains unchanged (0).
  + XOR gate 🡪 arcs are non unate

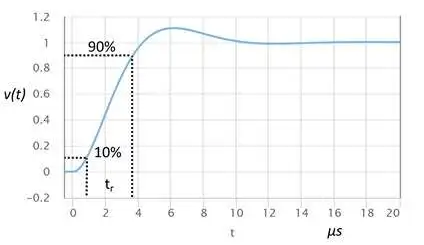
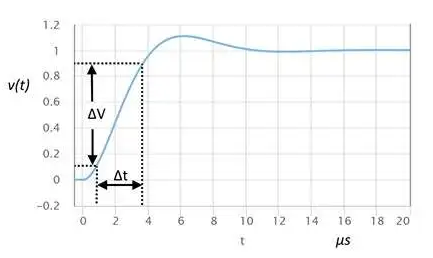
Consider the rising input signals:

* + - A is rising (0 🡪 1):
      * if B = 0, Q is rising (0 🡪 1);
      * if B = 1, Q is falling (1 🡪 0).
    - B is rising (0 🡪 1):
      * if A = 0, Q is rising (0 🡪 1);
      * if A = 1, Q is falling (1 🡪 0).

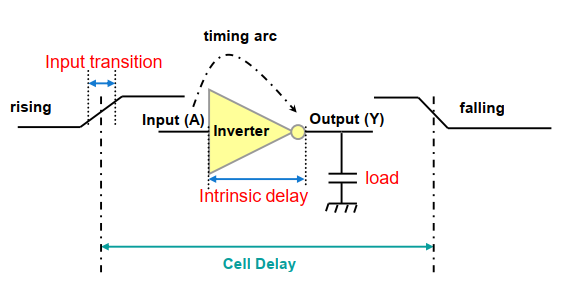
Consider falling input signals:

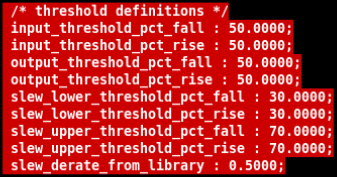
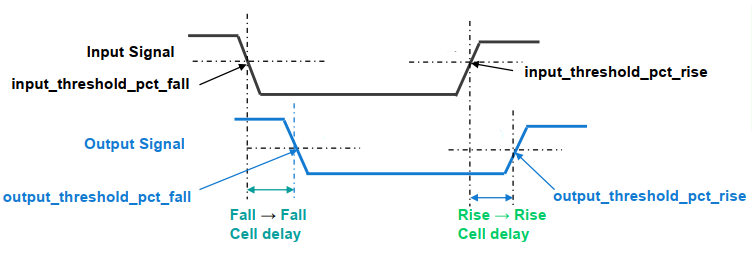
* + - A is falling (1 🡪 0):
      * if B = 0, Q is falling (1 🡪 0);
      * if B = 1, Q is rising (0 🡪 1).
    - B is falling (1 🡪 0):
      * if A = 0, Q is falling (1 🡪 0);
      * if A = 1, Q is rising (0 🡪 1).

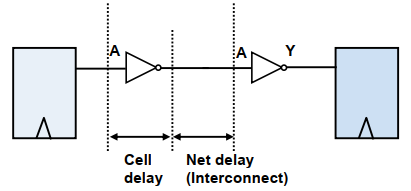
***Transition time & Slew***

Slew and transition times are sometimes used interchangeably, depending on the context. By definition, transition time refers to the time taken for a signal to transition between two specific levels, whereas slew (or slew rate) concerns with the rate of change of the output voltage, or the speed of transition. In this sense, the transition time is actually inverse of the slew rate – the larger the transition time, the slower the slew, and vice versa.

***Delay = cell delay + net delay***

**Cell delay:** Cell / propagation delay is the total delay between when the input signal changes state to when the output changes state. This delay value is dependent on the intrinsic delay of the cell, the load it’s driving, and the input transition times.

The propagation delays are measured from the input threshold percentage point (a.k.a input crossing its threshold point, typically 50% of input signal) to the output threshold percentage point (50% of the output signal).

**Net delay:** Net delay refers to an inherent delay between the time a signal is first applied to the net and the time it reaches other devices connected to that net. Basically, it is the time needed to charge or discharge all the parasitics of the net, i.e resistive, capacitive,…

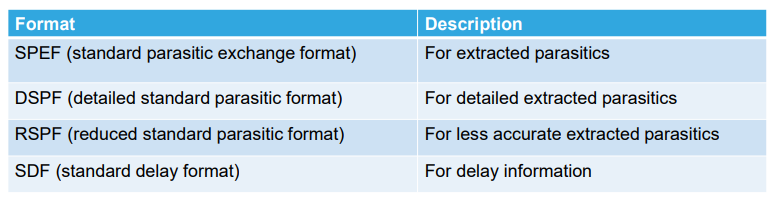
Since net delay is related to the physical properties of the wire, in pre-layout stages, we need to rely on some form of wire delay estimation to accurately predict the delays on the paths. Typically, wire delay estimation can be done using wire load model (WLM). Below are the steps of how STA tool may use WLM to calculate net delay:

1. Determines the **area** that a net fits in.
2. From the wireload selection table, **selects the WLM** with a block area that just encloses the area of the net.
3. Uses the WLM to **estimate the wire length** based on the fanout of the net or uses the wireload table to model capacitance and resistance more accurately.
4. The capacitance multiplier (CM) and resistance multipliers (RM) are used to determine the actual RCs of the net using the following:

▪ Cnet = CM\*length

▪ Rnet = RM \*length

1. The delay of the net is calculated using: ▪ Delay = Rnet\*Cnet

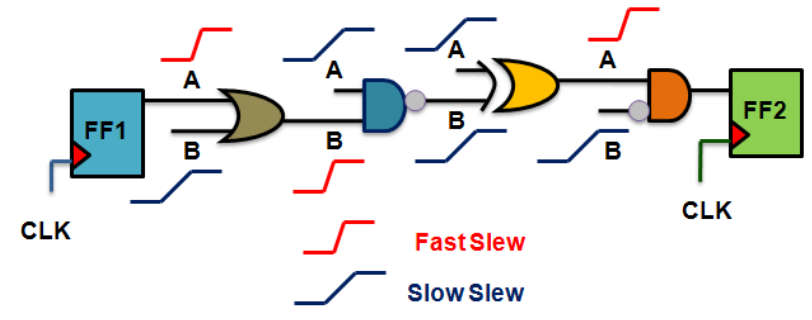
When the design is placed and routed, the extracted parasitic (RC) information is gathered. Extraction tools can create models for RC network using some sort of compression or reduction mechanism. These models are then reduced to become the RSPF or DSPF or the more commonly used SPEF files.

SPEF can be passed back to the synthesis tools either directly as parasitics or as delays (SDF) obtained from a delay calculator. This process is known as **back-annotation**.

***PBA – GBA***

In GBA mode, pessimistic transition time is propagated at each cell of the timing graph. GBA is able to calculate all the path delays in circuits and report the critical paths in a fast way. However, it always introduces pessimism due to the worst-case slew propagation.

As for PBA, timing analysis pessimism is reduced at the cost of significantly greater runtime than GBA. In PBA, the slew of the output pin is computed based on the real slew of inputs. Then the actual path-specific slew is propagated on the path and used in delay calculation for cells, which helps to improve the accuracy of time analysis. However, as the number of timing paths increases, there is an exponential increment in the possibilities of transition propagation and delay calculation at each cell. It causes PBA runtime-intensive.

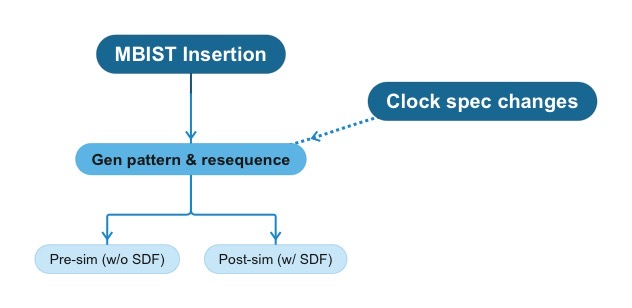
Consider the following example. Assume that we are doing setup check (max delay) for the timing path from FF1 to FF2.

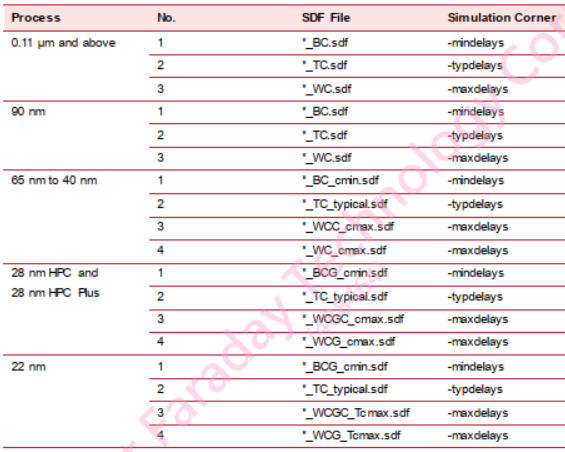
In GPA, slews propagated to the outpin of each gates will be the worst, meaning it will be computed based on the gates’ output load and worst input slews, namely:

* Slew at pin B to Z for ◼
* Slew at pin A to Z for ◼
* …

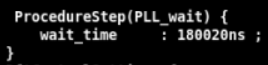
In PBA, the tool will consider the actual slew for the arcs encountered while traversing any particular timing path.

* Slew at pin A to Z for ◼
* Slew at pin B to Z for ◼
* …

***MBIST Simulation & Debug***

The **corners** used in simulation for different processes are not the same. Below are the suggested:

Notable **checkpoints** in debugging simulation:

* General:
  + Clock frequency match with spec and there is clock signal fed into MMEM.
  + Verilog models are read sufficiently.
  + Clock going through non-test MMEMs are off.
  + Clock signal is stable before entering MBIST pattern.
* Pre-sim: add\_seq\_delay should be larger than timescale unit.
* Post-sim:
  + SDF annotated successfully (> 90%) and all corners follow process signoff.
  + STA result of related MMEM / controller is met, or violated with a slack value < predefined uncertainty.
  + Asynch FFs timing are waived (by sourcing \*\_ann.tcl in SDF gen).
  + (If failed due to timing violation relating to divider) Divider(s)’ timing are waived.